Research Journal of Information Technology, 2011
ISSN 1815-7432 / DOI: 10.3923/rjit.2011.

A Review of Software Component Reusability Assessment Approaches

Fazal-e-Amin, Ahmad Kamil Mahmood and Alan Oxley
Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, 31750, Tronoh Perak, Malaysia

Corresponding Author: Fazal-e-Amin, Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, 31750, Tronoh, Perak, Malaysia

ABSTRACT

Software reuse yields benefits such as a reduction in the development time, cost and effort required and an increase in the productivity and quality. One of the ways to attain these reuse benefits is the use of components to develop software and is termed component-based software development. Reuse is facilitated by employing a systematic reuse methodology, such as is used with software product lines. Reusability is the degree to which software can be reused. Several approaches to assessing the reusability of software components have been proposed. Each is applicable to a specific programming language or paradigm. This study discusses the current state of the art of the reusability assessment of software components. The results of the review are categorized on the basis of the type of approach, the applicability and type of validation used to validate the approach. Moreover, the work attempts to highlight the applicability of available reusability assessment approaches with a view to identifying research gaps in this area.

Key words: Software reusability assessment, software component, review

INTRODUCTION

The term software engineering came into common usage by the computing community after the first NATO software engineering conference in 1968. The IEEEs standard glossary of software engineering terminology (IEEE Press, 1990) defines it as "The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software"; that is, the application of engineering to software.

The development of high quality software at an affordable cost and within a reasonable time scale is always desired by the software industry. Software reuse, being the process of creating new software by using existing software artefacts, is one of the ways to achieve the above mentioned goals. The reuse of software artefacts contributes to enhancing the quality of systems. It reduces the time and effort to develop and maintain new systems (Krueger, 1992). Evidence of improved quality by employing reuse is provided in Frakes and Succi (2001), where the relationship between amount of reuse, quality and productivity is explored. It is concluded, on the basis of four sets of industrial data gathered, that more reuse results in better quality. In a review of industrial studies to explore the effects of reuse on productivity and quality (Mohagheghi and Conradi, 2007), it is stated that reuse significantly reduces correction efforts and thus increases productivity.

Software reusability is defined as the property of software relating to the probability of reusing it (Frakes and Kyo, 2005). It can also be said to be the degree to which a software module or other

work product can be used in more than one computer program or software system (IEEE Press, 1990). There is a need to treat software reuse as an empirical discipline, so that it can be based on science and engineering (Frakes and Kang, 2005). The reported work on the benefits of reuse and the establishment of the concepts of reuse and usability compels us to expend effort in its measurement (Frakes and Kang, 2005). These measurement efforts are categorized into six types, which are: the reuse cost-benefits model; maturity assessment; the amount of reuse; the failure modes; the reusability metrics; the reuse library metrics (Frakes and Terry, 1996). Reusability assessment is considered as an important area in reuse measurement (Frakes and Terry, 1996). Reusability is an external attribute of software. An external attribute is one that cannot be measured directly. In contrast, internal attributes can be measured directly. If we can measure something directly then this means that we can measure it independently (IEEE press, 1998). For example, the size of a program can be measured directly in several ways: by counting the number of lines of code; by counting the number of methods. In software engineering measurement terminology, a metric is a quantitative indicator of a software attribute; a metrics model specifies relationships between metrics and the attributes being measured by these metrics.

Goulao and Abreu (2004) presented an overview of metrics-based software component quality evaluation. The evaluation of several approaches are presented and categorized. The focus of this study was overall quality assessment. By comparison, our scope is relatively limited. It concerns only approaches proposed to assess the reusability of software components.

Research methodology: A systematic literature review process is presented in Brereton *et al.* (2007) and the authors have presented their experiences regarding the reviews in software engineering. In our work we have followed an adapted form of this systematic review process. The process we followed is depicted in Fig. 1.

The review process has three phases and ten sub activities. In the first phase of the review, the following questions are posed:

- **Question 1:** What approaches have been introduced to assess software component reusability?

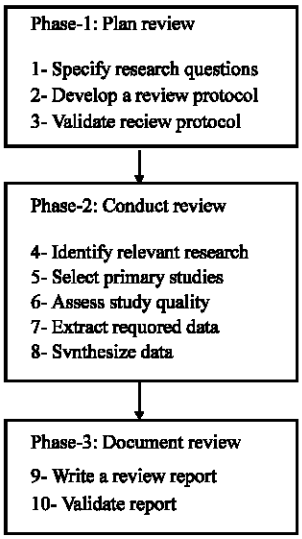

Fig. 1: Review process-adapted from Brereton *et al.* (2007)

Table 1: Review protocol

Year	Sources	Key words
2000-2010	Scopus, Google scholar, IEEE Xplore, ScienceDirect, ACM portal digital library	Software reusability evaluation/ assessment/ measurement, Software component reusability metrics/ model/ framework

- **Question 2:** What is the applicability of these approaches?
- **Question 3:** What is the procedure adopted for validating the approach?

After specifying the questions, a review protocol is developed. The review protocol includes the sources used in the review, the period of time under review and the search criteria used. This protocol is reviewed by the researchers and after making some changes it is validated.

The final review protocol is shown in Table 1. The sources used for this survey are Scopus, Google scholar, IEEE Xplore, ScienceDirect and the ACM portal digital library. The search focuses on the years 2000 to 2010.

In the second phase of the review, the search is performed by using different queries relating to the reusability assessment of software components. The initial collection of research papers is refined on the basis of trying to find the key words in Table 1 in the papers keywords and abstracts. A paper is to be included if it contains either a model, or a framework, or metrics for reusability assessment and there is also a demonstration of the applicability of the proposed solution with results. The required data to answer the questions posed above is extracted from the papers.

A table is formed to extract information/data from the selected papers. The table contains the following columns: Year, Complete Reference, Proposal, Applicability, Application/Demonstration and Validation. The construction of this table helped to organise the extracted information/data with the aim of providing a clear picture of the works in this area. Furthermore, the table serves to give the bare bones of a review of each paper. The information about the problem being highlighted and solved, in the paper is not included in the table because the papers were initially screened, leaving only those that are concerned with the reusability assessment of components.

Another step in the search process is performed by searching the related works sections of the collected papers. This step helps to enhance the strength of the review by ensuring that no valuable reference is missed during the search process.

The data is synthesized to exhibit comprehensive results. Finally, in the third phase of the review, the review report is documented and validated by the researchers.

RESULTS

The results of the review are presented in this section. A year wise representation of the results is presented in Table 2. The results are organized in the following sections according to the questions posed earlier.

A recent systematic review of software engineering measurements (Gomez *et al.*, 2008) reveals that few of the studies attempted to measure reusability. In our work, we managed to locate more studies that attempt to measure reusability. This may be due to the fact that we have used two resources that Gomez *et al.* (2008) have not the Scopus repository and the Google Scholar search engine.

Question 1: What approaches have been introduced to assess software component reusability?: The results of the review (Fig. 1) show the proposed approaches to assess the

Table 2: Year wise search results

Year	No. of publications
2000	0
2001	2
2002	1
2003	4
2004	1
2005	2
2006	1
2007	1
2008	4
2009	3
2010	1
Total	20

Table 3: Categories wise results of Question 1

Question	Category	No. of papers
What approaches have been introduced to assess	Hierarchical model and metrics	4
software component reusability?	Quality model including reusability	3
	Process	2
	Metrics	8
	Guidelines	1
	Framework	1
	Neural Network-based approach	1
	Total	20

reusability of a software component. These results are categorised into: (i) Hierarchical model and metrics, where a hierarchical model is presented which shows the relationship of reusability to its attributes. In some approaches this relationship is defined to the level of sub attributes. The model is companioned with metrics to calculate the values of attributes; (2) Quality model that includes reusability as a quality factor. Reusability may not be the only factor being considered; (3) Metrics. Some of the studies present a proposal for using metrics to assess reusability; (4) Process, A few papers have presented a process to assess reusability; (5) Guidelines. Some of the studies have outlined some guidelines for reusability; (6) Framework and (7) Neural network-based approach. The results are summarized in Table 3.

The results show that the most common proposed approach was to define metrics (40%) to assess reusability. In Cho *et al.* (2001), metrics to measure complexity, customizability and reusability are proposed. The degree of features reused in developing an application is used to measure reusability. Two types of metrics are proposed, one is the metrics to be used at the design phase and the second is the metrics used after coding - the number of lines of code; the proportion of overall functionality that each component has. The application of the proposed approach is demonstrated by applying it to components in the banking domain. A set of metrics to measure understandability and reusability of software components is presented by Boxall and Araban (2004) and applied to the measurement of twelve components. Rotaru and Dobre (2005) discussed the adaptability and complexity of software components. Compos- ability and adaptability are considered as main factors influencing reusability and metrics are presented to measure these factors. Two metrics are

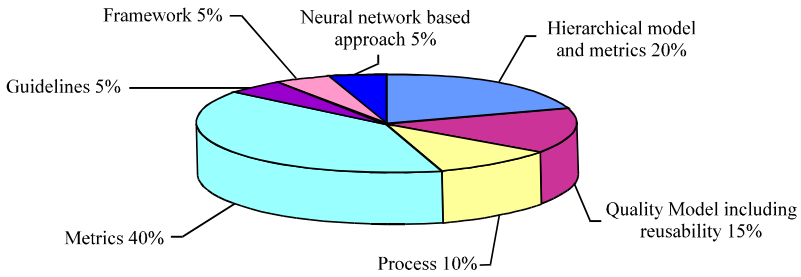

Fig. 2: Proposed approaches to assess reusability

proposed Aggarwal *et al.* (2005) to measure the amount of generic code. The proposed metrics are applied to ten projects. Gui (2006), defined measures for cohesion to assess the reusability of a component; the specific purpose is the prediction of effort required to reuse a component in a larger system. The proposed metrics are applied to a HTML parser, a lexical tokenizer and a bar code application, all of which are in the form of components. Gui and Scott (2007) proposed coupling metrics to rank the reusability of components. Metrics are applied to three types of component to generate the results. The metrics to measure coupling Gui and Scott (2007) and cohesion (Gui, 2006) are combined in Gui and Scott (2009) to measure the reusability of software components. In Gui and Scott (2008), coupling and cohesion metrics are also proposed to evaluate the reusability of components.

In our previous work (Husein and Oxley, 2009), a coupling and cohesion metrics suite is presented to evaluate object-oriented software. These metrics may be applied to assess reusability.

Hierarchical models and metrics (20%) are proposed which associate the factors and sub factors of reusability and metrics are defined to measure the lower level factors to assess reusability (Fig. 2). Etzkorn *et al.* (2001) defined three views of reusability which are: reusability in class, reusability in a hierarchy/ subsystem and reusability in the original system. Factors, sub factors and metrics are proposed to measure reusability. The proposed solution is applied to graphical user interface packages to generate the results. In Dandashi (2002), adaptability, completeness, maintainability and understandability are considered as factors affecting reusability. These factors are measured by the metrics. These metrics are applied to the components of a scientific application in order to evaluate the approach. A metric suite to measure reusability in presented in Washizaki *et al.* (2003). Metrics are applied on components available on the Web. In SantAnna *et al.* (2003), a framework is presented that contains a reusability and maintainability model and metrics for aspect oriented software. The proposed approach is applied on the aspect oriented implementation of GoF design patterns.

A recent review Fazal-e-Amin *et al.* (2010a) reveals that there is a growing trend in research towards the aspect oriented implementation of product line components. Software product line development is reuse intense development, in which software assets are developed by using existing assets. By combining the results of study (Fazal-e-Amin *et al.*, 2010a) with the current review, it can be observed that there is a research gap in reusability assessment of aspect oriented components. As part of on-going work, an assessment framework for reusability is presented in Fazal-e-Amin and Mahmood (2009) and a reusability attribute model for aspect oriented product line components is proposed in Fazal-e-Amin *et al.* (2010b). This model explores the attributes affecting the reusability of aspect oriented components.

Some of the results of the review contain software quality models including reusability as a quality factor. These include Kim and Park (2003), in which the quality characteristics of COTS components are identified in order to build a quality model. Commonality, customizability, modularity and comprehensiveness are defined as measures of reusability. In Yoonjung *et al.* (2008), a component quality model is presented which includes reusability as quality factor. The model is applied to applications in the digital TV domain. A quality model is presented in Kalaimagal and Srinivasan (2010), with reusability as a quality factor. The model is accompanied by metrics; it is applied on two software projects to generate the results.

The results of our review show that 10% of the studies used a measurement processess. Noor *et al.* (2008) proposed a collaborative scoping approach for an organization to migrate existing products to a product line. This approach makes use of metrics to assess reusability in one of its tasks. These metrics are applied at class, method and lines of code levels. A reusability measurement process is proposed in Bi *et al.* (2009); it uses McCabe and Halstead methods to measure reusability.

Guidelines, neural network-based and framework approaches each represent 5% of the results. In Gill (2003), guidelines are provided for reusability of software components, which can also be used to measure reusability. An artificial neural network-based approach to assess reusability is presented in Sharma *et al.* (2009). The network is trained with forty examples of Java and tested afterwards with twelve examples. A framework to measure and evaluate program source code is proposed in Washizaki *et al.* (2007); it contains a quality model and metrics. The model includes reusability as a quality factor. The implementation of the framework was demonstrated for use with software written in the C language and is restricted to this language.

Question-2: What is the applicability of these approaches? The applicability of the proposed approaches is categorized in three ways. First, the approaches are differentiated according to whether they are object oriented, aspect oriented, etc. (Fig. 3). Second, the approaches are categorised according to the language used to apply the approach -either C/C++ or Java. Third, the approaches are categorised according to whether the metrics used to assess reusability did so without reference to the source code of components, or by optionally referring to the source code, or by compulsorily referring to the source code. The results are presented in Table 4.

Figure 3 shows that 70% of proposed approaches are object oriented; they make use of the constructs of object orientation to measure reusability. Figure-4 further categorizes the results according to whether the approach was based on Java or C/C++. Reusability assessment approaches applicable to Java are 71% while 29% of the approaches are applicable to C/C++.

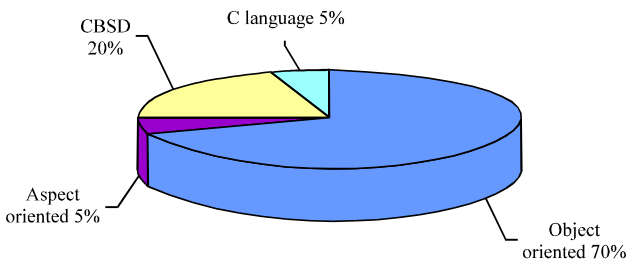

Fig. 3: Orientation of proposed approaches

Table 4: Categories wise results of question 2

Question	Category	No. of papers
What is the applicability of these approaches?	Object oriented	14
	Aspect oriented	1
	CBSD	4
	C Language	1
	Total	20
	C++	4
	Java	10
	C	1
	Total	15*
	White box	9
	Black box	5
	White and black box	1
	Total	15*

* Some of the publications cannot be categorized according to this breakdown

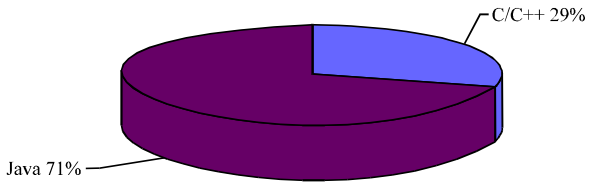

Fig. 4: Language-based classification

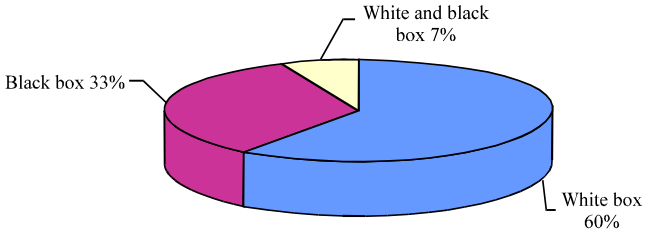

Fig. 5: Role of source code

Figure 5 reveals that 63% of the proposed approaches are white box; i.e. the availability of source code is mandatory with these approaches. 31% of approaches are black box, i.e. where source code is not referred to. The approach proposed in Etzkorn *et al.* (2001) can be used for both white and black box components.

Question-3: What is the procedure adopted for validating the approach? Figure 5 shows the results of the review as regards to the procedure adopted for validation. The categories are: (1) Human evaluation, where the results obtained by the application of proposed assessment approach is compared against the results gathered by the user/expert assessment of the component; (2) Statistical analysis, where the results are analyzed by using some statistical technique; (3) A few have applied the Weyukers properties to validate; (4) Experiment, where an experiment is carried out to validate the results; (5) Case studies are also used to validate the results; (6) Questionnaire

is used to assess the component and then the results of the approach are compared against the results collected from the questionnaire; (7) Test data is used to validate the assessment approach; (8) Some of the proposed approaches have not performed any validation. The details are presented below; Table 5 provides numbers for each category. Let us clarify the term validation. It refers to the any kind of empirical method used as a proof, apart from the demonstration/ application of the proposed approach.

The results of the question regarding the validation of proposed approaches shows that 30% of the selected papers proposed an approach to assess the reusability but provide no validation of the proposed approach. Statistical analysis of the results are provided in 25% of the studies, which include (Boxall and Araban, 2004). It uses linear regression and the mean to perform the statistical analysis. In Gui (2006), linear regression is used to evaluate the performance of the proposed measures of reusability. Rank correlation and linear regression is used to assess the performance of the proposed metrics in Gui and Scott (2007, 2008, 2009).

Case studies are used to validate the proposed approach in 10% of the selected papers, which include Yoonjung *et al.* (2008) where the case study of a digital TV application and related platform application is used. In Noor *et al.* (2008) the initial validation of the proposed approach is demonstrated through two open source projects.

Human evaluations as the form of validation are employed in 10% of the selected studies Fig. 6. Here the qualitative validation refers to a comparison between the results collected by applying the proposed approach and human evaluation of the components; these evaluators may include experts and users/software engineers. Etzkorn *et al.* (2001) collected expert opinion about the reusability of the components and then compared with results generated through the

Table 5: Categories wise results of question 3

Question	Category	No. of papers
What is the procedure adopted for validating the approach?	Human evaluation	2
	No validation	6
	Statistical Analysis	5
	Weyukers properties	1
	Experiment	1
	Case study	2
	Questionnaire	2
	Test data	1
	Total	20

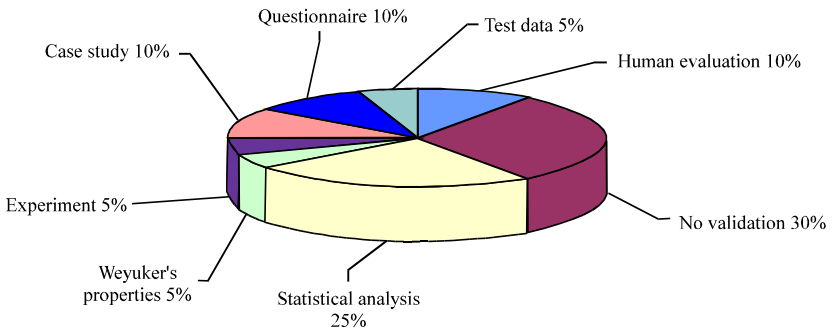

Fig. 6: Type of validation

application of the proposed approach; regression analysis is used to interpret the results. Washizaki *et al.* (2003) used the ratings of an evaluation committee of a web site, from where the components were selected, to assess their reusability.

Questionnaires are used in 10% of the studies. In Dandashi (2002), the correlation coefficient between results of direct measure using the proposed approach results and measures collected via the survey instrument, are presented to validate the results. Washizaki *et al.* (2007) used a questionnaire to validate the framework using quantitative evaluation.

Aggarwal *et al.* (2005) Weyukers properties are used to evaluate the metrics. SantAnna *et al.* (2003) performed a semi-controlled experiment with human subjects who implemented two versions of a web portal development system. Sharma *et al.* (2009) proposed a neural network-based approach to assess the reusability and it is validated by using test data.

Conclusions and future directions: A literature review of the works in the area of software reusability assessment is conducted and the results of the review are presented in this paper. The results show that the majority of the approaches are based on metrics (70%), are applicable to the object oriented paradigm (70%) and the target language used for the application is Java (71%). These results point towards the fact that the interest of the software development community is in object oriented java-based development. The results show a lack of validation in the proposed approaches as 30% of the studies provide no validation of the results. Only a few studies have used experimentation to validate. This area (validation of defined measures) needs the attention of the research community to gain the confidence of the software practitioners. A majority of approaches (60%) are white box-based. These approaches can be used to measure open source components. This shows the trend of component-based software development is towards using open source software components.

Although our review has explored the field, further studies are needed to confirm the obtained results. Future work includes the extension of this review by including more sources (conferences and workshops) and questions. A future direction is the inclusion of more questions to explore the other dimensions, such as the domain wise categorization of the approaches.

REFERENCES

Aggarwal, K.K., Y. Singh, A. Kaur and R. Malhotra, 2005. Software reuse metrics for object-oriented systems. Proceedings of the 3rd ACIS International Conference on Software Engineering Research, Management and Applications, Aug. 11-13, IEEE Computer Society, Washington, DC, USA., pp: 48-55.

Bi, S., X. Dong and S. Xue, 2009. A measurement model of reusability for evaluating component. Proceedings of the 1st IEEE International Conference on Information Science and Engineering, Dec. 26-28, IEEE Computer Society, Washington, DC, USA., pp: 20-22.

Boxall, M.A.S. and S. Araban, 2004. Interface metrics for reusability analysis of components. Proceedings of the Australian Software Engineering Conference, April 13-16, IEEE Computer Society, Washington, DC, USA., pp: 40-51.

Brereton, P., B.A. Kitchenham, D. Budgen, M. Turner and M. Khalil, 2007. Lessons from applying the systematic literature review process within the software engineering domain. J. Syst. Softw., 80: 571-583.

Cho, E.S., M.S. Kim and S.D. Kim, 2001. Component metrics to measure component quality. Software engineering conference. Proceedings of the 8th Asia-Pacific on Software Engineering Conference Dec. 04-07, IEEE Computer Society, Washington, DC, USA., pp: 419-426.

Dandashi, F., 2002. A method for assessing the reusability of object-oriented code using a validated set of automated measurements. Proceedings of the ACM Symposium on Applied Computing, Madrid, Spain, March 11-14, ACM, New York, USA., pp: 997-1003.

Etzkorn, L.H., W.E. Hughes and C.G. Davis, 2001. Automated reusability quality analysis of OO legacy software. Inform. Software Technol., 43: 295-308.

Fazal-e-Amin and A.K. Mahmood, 2009. A survey and proposed reusability assessment framework for aspect oriented product line core assets. Proceedings of the IEEE Student Conference on Research and Development, Nov. 16-18, UPM Serdang, pp: 192-195.

Fazal-e-Amin, A.K. Mahmood and A. Oxley, 2010a. A proposed reusability attribute model for aspect oriented software product line components. Proceedings of the International Symposium on Information Technology, June 15-17, Kuala Lumpur, Malaysia, pp: 1138-1141.

Fazal-e-Amin, A.K. Mahmood and A. Oxley, 2010b. A review on aspect oriented implementation of software product lines components. Inform. Technol. J., 9: 1262-1269.

Frakes, W. and C. Terry, 1996. Software reuse: Metrics and models. ACM Comput. Surv., 28: 415-435.

Frakes, W.B. and G. Succi, 2001. An industrial study of reuse, quality and productivity. J. Syst. Softw., 57: 99-106.

Frakes, W.B. and K. Kang, 2005. Software Reuse research: Status and future. IEEE Trans. Software Eng., 31: 529-536.

Gill, N.S., 2003. Reusability issues in component-based development. SIGSOFT Softw. Eng. Notes, 28: 4-4.

Gomez, O., H. Oktaba, M. Piattini and F. Garcia, 2008. A systematic review measurement in software engineering: State-of-the-art in measures. Software Data Technol. Commun. Comput. Inform. Sci., 10: 165-176.

Goulao, M. and F.B. Abreu, 2004. Software components evaluation: An overview. Proceedings of the 5th Conferencia da APSI, (CA04), Lisbon, pp: 1-12.

Gui, G., 2006. Component reusability and cohesion measures in object-oriented systems. Proceedings of the 2nd Information and Communication Technologies, (ICT06), Damascus, pp: 2878-2882.

Gui, G. and P.D. Scott, 2007. Ranking reusability of software components using coupling metrics. J. Syst. Software, 80: 1450-1459.

Gui, G. and P.D. Scott, 2008. New coupling and cohesion metrics for evaluation of software component reusability. Proceedings of the 9th International Conference for, Young Computer Scientists, Nov. 18-21, IEEE Computer Society, Washington, DC, USA., pp: 1181-1186.

Gui, G. and P.D. Scott, 2009. Measuring software component reusability by coupling and cohesion metrics. J. Comput., 4: 797-805.

Husein, S. and A. Oxley, 2009. A coupling and cohesion metrics suite for object-oriented software. Proceedings of International Conference on Computer Technology and Development, Nov. 13-15, Kota Kinabalu, Malaysia, pp: 421-425.

IEEE Press, 1990. IEEE Standard Glossary of Software Engineering Technology. ANSI/IEEE Standard, Washington, DC, USA.

IEEE Press, 1998. IEEE Standard for a Software Quality Metrics Methodology. Institute of Electrical and Electronics Engineers, IEEE 1061-1998, USA., ISBN: 0738110596, pp: 32.

Kalaimagal, S. and R. Srinivasan, 2010. Q FACTO 10-A commercial off-the-shelf component quality model proposal. J. Software Eng., 4: 1-15.

Kim, S.D. and J.H. Park, 2003. C-QM: A practical quality model for evaluating COTS components. Proceedings of the IASTED International Conference on Software Engineering, Innsbruck, Austria, Feb. 10-13, IASTED/ACTA Press, pp: 991-996.

Krueger, C.W., 1992. Software reuse. ACM Comput. Surv., 24: 131-183.

Mohagheghi, P. and R. Conradi, 2007. Quality, productivity and economic benefits of software reuse: A review of industrial studies. Empirical Software Eng., 12: 471-516.

Noor, M., P. Grunbacher and C. Hoyer, 2008. A collaborative method for reuse potential assessment in reengineering-based product line adoption. Balancing Agility Formalism Software Eng., 5082: 69-83.

Rotaru, O.P. and M. Dobre, 2005. Reusability metrics for software components. Proceedings of the 3rd ACS/IEEE International Conference on Computer Systems and Applications, Jan. 3-6, IEEE Computer Society, Washington, DC, USA., pp: 24-I-24-I.

SantAnna, C., A. Garcia, C. Chavez, C. Lucena and A. von Staa, 2003. On the reuse and maintenance of aspect-oriented software: An assessment framework. Proceedings of the XVII Brazilian Symposium on Software Engineering, (BSSE03), UFBA, Computer Science Department, pp: 19-34.

Sharma, A., P.S. Grover and R. Kumar, 2009. Reusability assessment for software components. ACM SIGSOFT Software Eng. Notes 34: 1-6.

Washizaki, H., H. Yamamoto and Y. Fukazawa, 2003. A metrics suite for measuring reusability of software components. Proceedings of the 9th International Symposium on Software Metrics, Sept. 3-5, Sydney, Australia, pp: 211-211.

Washizaki, H., R. Namiki, T. Fukuoka, Y. Harada and H. Watanabe, 2007. A framework for measuring and evaluating program source code quality. Proceedings of the 8th International Conference on Product-Focused Software Process Improvement, Riga, Latvia, July 2-4, Springer-Verlag, Berlin, Heidelberg, pp: 284-299.

Yoonjung, C., L. Sungwook, S. Houp and S.H. Kim, 2008. Practical S/W component quality evaluation model. Proceedings of the 10th International Conference on Advanced Communication Technology, Feb. 17-20, Gangwon-Do, pp: 259-264.